เขียนโปรแกรมใช้เอง ดีอย่างไร และ ต้องระวังอะไรบ้าง

ดร.ปัญญา เปรมปรีดิ์

เกริ่นกันก่อน

หลาย ๆ คนถามความเห็นของผมอยู่เสมอว่า เราควรจะให้ยูสเซอร์เขียนโปรแกรมเองดีหรือไม่ และนโยบายที่ใช้อยู่ในหน่วยงานที่ผมทำงานอยู่นั้นเป็นอย่างไร เรื่องนี้ผมจึงตอบไปว่า *"เราควรสอนให้คนของยูสเซอร์ทุกคนรู้เรื่องคอมพิวเตอร์ และถ้าคนไหนมีความสามารถพอที่จะเขียนโปรแกรมขึ้นใช้เองได้เราก็ควรส่งเสริม แต่ไม่ควรให้เขาตั้งส่วนงานโปรแกรมมิ่งขึ้นมา"*

ครับ...ฟังแล้วก็คงจะงง ๆ กันอยู่ ความหมายของผมก็คือ

"ส่งเสริมให้คนในฝ่ายงานต่าง ๆ เขียนโปรแกรมขึ้นมาใช้งานเองได้ แต่ต้องเขียนให้ตัวเองใช้"

ผมไม่สนับสนุนให้สร้างทีมพัฒนาโปรแกรมขึ้นมาใหม่ในฝ่ายงานต่าง ๆ เพราะเท่ากับเป็นการทำงานซ้ำซ้อนกับฝ่ายคอมพิวเตอร์ และยังเป็นวิธีที่ไม่มีประสิทธิภาพ

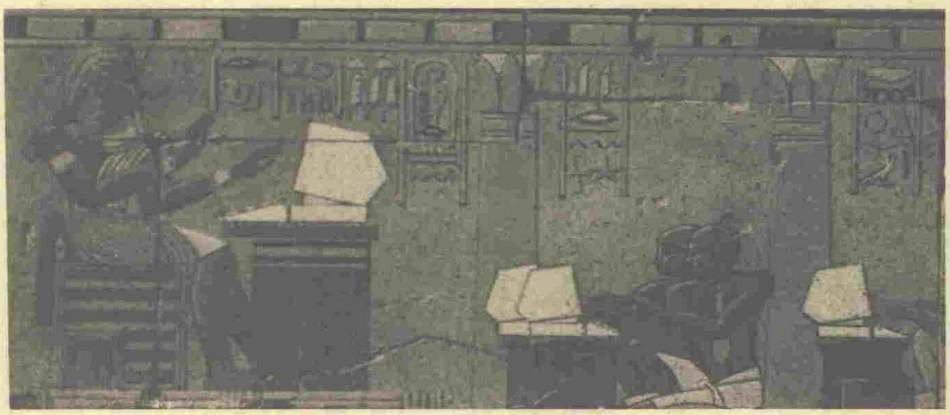
การที่ผมมีแนวคิดดังกล่าวนี้สืบเนื่องจากผมเล็งเห็นว่ามันมีความแตกต่างกันระหว่างการเขียนโปรแกรมให้ตนเองใช้กับการเขียนโปรแกรมให้คนอื่นใช้ การเขียนโปรแกรมใช้เองนั้นง่ายกว่าและมีทางเป็นไปได้สำหรับคนในฝ่ายต่าง ๆ แต่การเขียนโปรแกรมให้คนอื่นใช้จะยากกว่า เพราะเป็นงานในระดับของผู้เชี่ยวชาญ จึงต้องให้คนในฝ่ายคอมพิวเตอร์ทำการตั้งทีมเล็ก ๆ ขึ้นมาในฝ่ายยูสเซอร์เพื่อพัฒนาโปรแกรมให้คนในฝ่ายของตัวเองใช้นั้น จะเกิดช่องโหว่ได้มาก และกำลังเป็นปัญหาอยู่ในองค์กรขนาดใหญ่ ๆ หลายแห่ง

คราวนี้เราลองมาดูความแตกต่างระหว่างการเขียนโปรแกรมขึ้นใช้เองกับการเขียนโปรแกรมให้คนอื่นใช้ แล้วเราจะได้ชี้จุดที่ควรระวังเมื่อยูสเซอร์ต้องการเขียนโปรแกรมขึ้นมาใช้งานเอง

เมื่อผู้ใช้เป็นโปรแกรมเมอร์ซะเอง

การใช้โปรแกรมหมายถึงการไปนั่งที่ตรงหน้าเครื่องคอมพิวเตอร์หรือเทอร์มินัล แล้วสั่งให้เครื่องนำโปรแกรมที่เขียนและแปลไว้แล้วออกมาทำงาน เครื่องจะไปอ่านโปรแกรมที่ต้องการมาเก็บไว้ในส่วนความจำ (Main Memory) แล้วเริ่มทำงานตามคำสั่งต่าง ๆ ภายในโปรแกรมนั้น โปรแกรมดังกล่าวอาจมีการโต้ตอบหรือเรียกหาข้อมูลจากเทอร์มินัล ซึ่งผู้ใช้จะต้องคีย์ข้อมูลเข้าไปทางคีย์บอร์ด โปรแกรมอาจเรียกหาแผ่นจานดิสเกตต์เพื่อบันทึกข้อมูล ผู้ใช้ก็ใส่แผ่นดิสเกตต์...ฯลฯ

จากที่กล่าวมาข้างต้นเราจะเห็นว่า ถ้าเรารู้ขั้นตอนของโปรแกรม เราจะใช้มันได้ง่ายมาก ในทางตรงกันข้ามหากเราไม่รู้ขั้นตอนของโปรแกรมอาจก่อให้เกิดความสับสนได้ ยกตัวอย่างเช่น ถ้าในโปรแกรมใช้คำสั่งว่า INPUT A



เครื่องมันจะมาหยุดรอรับข้อมูลเพื่อนำไปใช้เป็นตัวแปร A ถ้าผู้ใช้ไม่รู้ขั้นตอนการทำงานเขาย่อมไม่ทราบเลยว่า เครื่องต้องการอะไร เพราะเครื่องจะมีอาการ "หยุดไปเฉย ๆ" ถึงตอนนี้ต่างคนต่างก็รอกัน หากผู้ใช้โปรแกรมเป็นคนประเภทขี้สงสัยอาจจะกดแป้นบางอันดูเล่น เช่น เคาะ space bar หรือกดแป้น enter ผลที่ตามมาก็คือ โปรแกรมจะรับค่า A เป็นเลขศูนย์ (0) เข้าไป แล้วทำงานต่อไปโดยผู้ใช้ไม่รู้ตัว การทำงานแบบนี้ไม่ช้าจะเกิดความสับสนขึ้นได้ เพราะเราไม่รู้ว่าโปรแกรมวิ่งไปถึงจุดไหนแล้ว ซ้ำร้ายข้อมูลที่รับเข้าไปยังเป็นข้อมูลผิด ๆ อีกด้วย

ปัญหาเรื่องเครื่องมาหยุดรอรับข้อมูลนี้แก้ไขได้ไม่ยาก คนเขียนโปรแกรมสามารถแสดงข้อความมายังจอภาพเสียก่อนที่จะถึงคำสั่งรับข้อมูล เช่น เปลี่ยนเป็น INPUT "A=" ; A ซึ่งหมายความว่า ให้แสดงข้อความ "A=" ขึ้นบนจอภาพก่อนที่จะหยุดรอรับค่าของ A แต่ก็ยังแก้ปัญหาได้ไม่หมด เพราะข้อความ "A=" นี้อาจเพียงพอสำหรับผู้เขียนโปรแกรมที่จะใส่ข้อมูล แต่ไม่ชัดเจนพอสำหรับคนอื่น ผู้อื่นที่ไม่ใช่ผู้เขียนโปรแกรมนี้เองจะถามว่า A ตัวนี้เป็นอะไร จะต้องใส่ตัวเลขแบบจำนวนเต็ม (Integer) ตัวเลขแบบมีทศนิยม (Floating point number) หรือข้อความ (String) ถึงแม้ว่าเราจะเปิดฝึกอบรมให้กับผู้ใช้โปรแกรมแล้ว ในเมื่อเขาไม่ใช่ผู้เขียนโปรแกรมนั้นเอง ก็อาจจะหลงลืมได้ ถ้าหากเขาใส่ข้อมูลผิดประเภทเข้าไปในเครื่องจะทำให้เกิดปัญหายุ่งยากหนักขึ้นอีก คือ โปรแกรมจะหยุดทำงานเลย เรื่องโปรแกรมหยุดทำงานเพราะข้อมูลผิดประเภทนี้เป็นเรื่องใหญ่ เพราะเราไม่รู้ว่าแฟ้มข้อมูลต่าง ๆ จะอยู่ในสภาพอย่างไร การย้อนกลับไปทำใหม่นั้นต้องให้ผู้เขียนโปรแกรมเป็นคนจัดการเอง ระบบแบบนี้จึงใช้ได้เฉพาะผู้เขียนโปรแกรมเองเท่านั้น

จะเขียนให้คนอื่นใช้ต้องเขียนอย่างไร

เราต้องเขียนโปรแกรมโดยป้องกันข้อผิดพลาดของผู้ใช้เอาไว้ให้หมด อย่างเช่น เรื่องคำสั่ง INPUT ที่กล่าวไปแล้ว เราจะใช้คำสั่งนี้กับผู้ใช้ที่ไม่ใช่โปรแกรมเมอร์ไม่ได้ เราต้องใช้คำสั่งชนิดอื่นมาแทน และมันจะซับซ้อนกว่าเดิมหลายสิบเท่า เพื่อให้ท่านผู้อ่านเข้าใจอย่างลึกซึ้งกับปัญหานี้ ผมก็จะขยายความให้ทราบ

ในชั้นต้นเราต้องแสดงข้อความที่ชัดเจนกว่าเก่า เช่น ถ้า A หมายถึงจำนวนนักเรียนในชั้นเรียน และต้องเป็นเลขจำนวนเต็ม ซึ่งมีค่าอยู่ระหว่าง 1 ถึง 60 เราอาจพิมพ์ข้อความว่า "PLEASE INPUT THE NUMBER OF STUDENTS (1 TO 60)" แล้วค่อยหยุดรอรับข้อมูล อย่างนี้จึงจะทำให้ผู้ใช้โปรแกรมพอจะทำงานได้ถูกต้อง คือเขาต้องมองหาข้อมูลจากจำนวนนักเรียนในชั้นนั้นมาคีย์ใส่เข้าไปทางคีย์บอร์ด อย่างไรก็ตามเราจะใช้คำสั่ง INPUT ในภาษาเบสิกมารับข้อมูลก็ยังไม่ได้ เพราะผู้ใช้อาจคีย์ตัวเลขปะปนกับตัวอักษรอื่น ๆ เข้ามาได้ หรือคีย์เป็นตัวเลขก็จริงแต่อาจอยู่นอกขอบเขต 1 ถึง 60 เข้ามา ซึ่งจะมีผลทำให้โปรแกรมทำงานผิดพลาดได้อีก เรื่องนี้ต้องแก้ไขในขั้นที่ 2

ขั้นที่ 2 คือต้องเลือกใช้คำสั่งที่เครื่องยอมรับได้ทุกครั้ง คำสั่งในภาษาเบสิกที่จะใช้ในการรับข้อมูลจากคีย์บอร์ดได้ทุกครั้งโดยไม่เกิด abnormal end ก็คือคำสั่ง INKEY$ คำสั่งนี้เป็นการดึงเอารหัสตัวอักษร 1 ตัว มาจากบัฟเฟอร์ (Buffer) ของคีย์บอร์ด แล้วนำมาใส่ในตัวแปรที่เราต้องการ ยกตัวอย่างเช่น ถ้าเราเขียนว่า X$ = INKEY$ ก็หมายความว่าตัวอักษรที่เรากดเข้าไปที่แป้นของคีย์บอร์ดจะถูกนำไปบรรจุลงที่ตัวแปร X$ (ตัวแปรชนิดที่มีเครื่องหมาย $ อยู่ท้ายนี้ในภาษาเบสิกจะเป็นประเภท String ซึ่งยอมรับได้หมดทุกแป้นบนคีย์บอร์ด) คำสั่งนี้ใช้ 1 ครั้งจะได้ 1 ตัวอักษร และการกดแป้นที่คีย์บอร์ดนั้นจะกดไว้ล่วงหน้าก็ได้ โดยเครื่องจะส่งมาให้ตามลำดับ ตัวอักษรใดกดก่อนก็ส่งมาให้ก่อน

ทีนี้สมมติว่าเราใช้คำสั่ง X$ = INKEY$ ไปแล้วหนึ่งครั้ง เราก็จะได้

รหัสตัวอักษร 1 ตัว ซึ่งบรรจุอยู่ในตัวแปร X$ ทีนี้ก็ต้องเช็คว่ารหัสตัวนี้เป็นประเภทตัวเลข "0" ถึง "9" หรือไม่ เพราะถ้าผู้กดแป้นพิมพ์ผิดเราจะได้เตือนให้ทราบ การเช็คนี้ก็ต้องใช้คำสั่ง IF เช่น IF (X$ < "0") OR (X$ > "9") THEN GOTO 100 นี่เป็นการถามว่ารหัส X$ นั้นน้อยกว่ารหัสเลข "0" หรือมากกว่ารหัสเลข "9" หรือไม่ ถ้าเป็นเช่นนั้นก็แสดงว่าเกิดความผิดพลาดขณะกดแป้นโปรแกรมจะไปที่คำสั่งที่ 100 เพื่อออกข้อความเตือนว่าผิดและให้คีย์ข้อมูลใหม่

พอถึงตรงนี้สมมติว่าผู้ใช้กดตัวเลขที่ถูกต้องเข้าไป X$ จึงเป็นรหัสที่มีค่าระหว่าง "0" ถึง "9" ในกรณีเช่นนี้จะไม่มีการกระโดดไปที่คำสั่ง 100 มันจะไหลต่อลงมา ดังนั้นคำสั่งถัดไปก็จะอยู่ใต้คำสั่ง IF ดังที่อธิบายให้ทราบแล้ว สิ่งที่เราต้องทำคือต้องเอา X$ นี้ไปเก็บไว้ที่ไหนสักจุดหนึ่ง สมมติว่าเราเก็บไว้ที่ A$ ค่าของ A$ นี้ในตอนเริ่มต้นเราควรเซตให้เป็น NULL หรือเขียนว่า A$ = " " (เครื่องหมายฟันหนู 2 ชุดติดกันคือ null string) และเมื่อได้ X$ มาแล้วก็เอามาต่อท้าย A$ ด้วยคำสั่ง A$ = A$+X$ ในภาษาอังกฤษเรียกว่า CONCATENATE แต่ใช้เครื่องหมายบวกเป็นตัวแสดงคำสั่ง ทำคำสั่งนี้แล้ว A$ ก็จะมีค่าเท่ากับ X$ ถ้าทำซ้ำอีกจะทำให้ A$ ยาวขึ้นอีกเรื่อย ๆ ทั้งนี้โดยมี X$ ตัวใหม่มาแนบต่อท้ายตลอดเวลา

ตกลงว่าเมื่อเก็บรหัสตัวเลขตัวแรกที่กดแป้นเข้าไปไว้ที่ A$ เรียบร้อยแล้วก็ต้องวนกลับไปใช้คำสั่ง X$ = INKEY$ และ IF (X$<"0") OR (X$>"9") THEN GOTO 100 ซ้ำอีก เท่ากับเป็นการรับเอาการกดแป้นครั้งที่ 2 เข้ามา หากโปรแกรมทำงานเป็นปกติดีเราจะได้ดิจิตที่ 2 แทน เราต้องนำมาต่อท้าย A$ ด้วยคำสั่ง A$ = A$+X$ อีก ถ้าเราวนอย่างนี้ไปเรื่อย ๆ เราจะได้ตัวเลขยาว n หลัก ซึ่งเก็บไว้เป็นสตริงอยู่ใน A$ เมื่อเราต้องการเปลี่ยนมาเป็นค่าของ A (A กับ A$ เป็นคนละตัวแปร A เป็นตัวเลข ส่วน A$ เป็นสตริง) เราก็ใช้คำสั่งว่า A = VAL (A$) คำสั่งนี้เปลี่ยนรูปจากสตริงมาเป็นไบนารีแล้วเก็บค่าไว้ที่ A

ครับ ยังไม่จบแค่นี้หรอก เพราะการคีย์ตัวเลขเข้าไปทางคีย์บอร์ดนี้ผู้พิมพ์อาจต้องการแก้ไขโดยกดแป้น Backspace นอกจากนี้แล้วตัวเลขอาจยาว

รูปที่ 1 ตัวอย่างรูทีนเพื่อรับค่าตัวเลขแทนคำสั่ง INPUT (A)

```
5    CLS
6    LOCATE 1,1
7    PRINT "PLEASE INPUT NUMBER OF STUDENTS (1 TO 60)";
8    A$ = "" : N = 0
10   X$ = INKEY$
11   IF X$ = "" THEN GOTO 10
12   IF X$ = CHR$ (8)  THEN GOTO 30
13   IF X$ = CHR$ (13) THEN GOTO 120
14   IF (X$ < "0") OR (X$ > "9") THEN GOTO 100
15   IF N = 2 THEN GOTO 110
16   N = N+1
17   A$ = A$+X$
18   LOCATE 1,42
19   PRINT A$;
20   GOTO 10
30   IF N = 0 THEN BEEP : LOCATE 1,42 : GOTO 10
31   LOCATE 1,42 : PRINT "  ";
32   N = N-1 : A$ = LEFT$ (A$,N) : LOCATE 1,42 : PRINT A$;
33   GOTO 10
100  LOCATE 2,1
101  PRINT "ERROR, NOT NUMERIC";
102  BEEP : LOCATE 1, 42 : PRINT "  "; : GOTO 6
110  LOCATE 2,1
111  PRINT "ERROR, TOO MANY DIGITS";
112  BEEP : LOCATE 1,42 : PRINT "  "; : GOTO 6
120  A = VAL (A$)
121  IF (A < 1) OR (A > 60) THEN GOTO 130 ELSE GOTO 200
130  LOCATE 2,1
131  PRINT "ERROR : INVALID VALUE"
132  BEEP : LOCATE 1,42 : PRINT "  "; : GOTO 6
200  ----------
```

เพียง 1 หลักก็เป็นได้ เช่น ชั้นเรียนนี้มีคนน้อยกว่า 10 คน กรณีแบบนี้ต้องมีการเช็คว่า X$ ที่รับเข้ามานั้นเป็นรหัส Backspace หรือ Enter หรือไม่ ถ้าใช่เราจำเป็นต้องดำเนินการไปอีกอย่างหนึ่ง เมื่อผู้พิมพ์กด Backspace ไปจนหมดแล้วยังกดซ้ำเข้ามาอีกก็ต้องบอกเป็นเสียงเตือน...ฯลฯ

กล่าวโดยสรุปแล้วจะเขียนโปรแกรมให้คนอื่นใช้นั้นต้องคิดลึกกว่าเดิมหลายสิบเท่า ในรูปที่ 1 เป็นการแสดงตัวอย่างการรับข้อมูลผ่านทางคีย์บอร์ดแทนคำสั่ง INPUT (A) ตามที่อธิบายมาแล้วข้างต้น

ปัญหาของผู้สร้างภาษาโปรแกรม

ผมเชื่อว่าผู้อ่านส่วนใหญ่ยังไม่ค่อยเข้าใจในรายละเอียดของตัวอย่างในรูปที่ 1 และผมไม่อยากให้ท่านต้องเสียเวลาทำความเข้าใจ เพราะมันไม่จำเป็นต้องเข้าใจ ท่านที่เป็นผู้บริหารหรือยูสเซอร์ที่ต้องการเขียนโปรแกรมเล็ก ๆ น้อย ๆ เพื่อใช้เองนั้น ใช้คำสั่งว่า INPUT "A=" ; A ก็เพียงพอแล้ว แต่สิ่งที่ผมอยากให้ท่านลองสังเกตสักนิดคือ ในภาษาที่จะใช้เขียนโปรแกรมต่าง ๆ นั้น เรามักจะเริ่มจากการตั้งคำสั่งขั้นพื้นฐานเอาไว้จำนวนหนึ่ง เช่น INPUT, PRINT, READ, WRITE, IF, GOTO, FOR และการบวกลบคูณหาร คำสั่งเหล่านี้ดู ๆ แล้วจะเพียงพอสำหรับใช้สั่งเครื่องคอมพิวเตอร์ให้ทำงานตามวัตถุประสงค์ของเราได้ เช่น ใช้ WRITE เพื่อเก็บข้อมูลไว้ในจานแม่เหล็ก เมื่อต้องการเรียกขึ้นมาเพื่อแก้ไขก็ใช้คำสั่ง READ เมื่ออ่านข้อมูลขึ้นมาได้แล้วก็ใช้คำสั่งด้านการบวกลบคูณหารมาทำการคำนวณ สำหรับข้อมูลที่จะใช้แก้ไขนั้นก็จะใช้คำสั่ง INPUT เพื่อรับข้อมูลเข้าทางแป้นคีย์บอร์ด พนักงานผู้ใช้โปรแกรมก็จะคีย์ข้อมูลเข้าไปได้ หลังจากที่นำข้อมูลจากคีย์บอร์ดเข้าไปแก้ไขเรคอร์ดนั้นแล้ว เราก็ใช้คำสั่ง WRITE เพื่อส่งข้อมูลที่แก้ไขเรียบร้อยกลับไปบันทึกลงจานแม่เหล็ก นอกจากนี้เรายังสามารถใช้คำสั่ง WRITE หรือ PRINT เพื่อแสดงผลเป็นรายงานบนจอภาพหรือพิมพ์ออกที่เครื่องพิมพ์ได้ สำหรับคำสั่ง IF นั้นใช้เพื่อการเลือกหนทางปฏิบัติ คำสั่ง GOTO มีไว้เพื่อใช้ข้ามคำสั่งหรือย้อนกลับ และคำสั่ง FOR ก็มีไว้เพื่อการวนเวียนหรือทำแบบเดิมหลาย ๆ ครั้ง จากคำสั่งขั้นพื้นฐานเหล่านี้เราสามารถสอนคนให้เขียนโปรแกรมได้ใน 2-3 วัน

"แต่เราจะพบว่าการเขียนโปรแกรมขึ้นมาเพื่อให้ใช้งานได้จริง ๆ และใช้โดยคนที่ไม่รู้เรื่องคอมพิวเตอร์นั้นไม่สามารถจะทำได้โดยคำสั่งพื้นฐานเหล่านี้"

เรื่องนี้เป็นปัญหาที่ผู้สร้างภาษาโปรแกรมรุ่นแรก ๆ ไม่เคยคาดคิดมาก่อน และผมยังเชื่อว่านักสร้างภาษาเพื่อใช้สั่งเครื่องรุ่นใหม่ส่วนใหญ่ก็ยังไม่เข้าใจปัญหา พวกเขามัวแต่สนใจเสาะแสวงวิธีที่จะทำให้การแปลภาษาระดับต่าง ๆ ไปสู่ภาษาเครื่อง (Machine Code) มีประสิทธิภาพสูงสุด อย่างไรก็ตามเราก็ยังมีคนที่ชอบเสนอให้เติมคำสั่งใหม่ ๆ เข้าไปในภาษาเก่า ๆ อยู่เสมอ เช่น เรามีคำสั่ง INKEY$, LEFT$, RIGHT$, MID$, VAL, CHR$ และการจัดการกับข้อความหรือสตริงเพิ่มเข้ามาในภาษาเบสิก ซึ่งเป็นหนทางออกของโปรแกรมเมอร์ เพราะพวกเขาจะได้อาศัยคำสั่งเหล่านี้เป็นตัวบังคับให้คนทำงานร่วมกับคอมพิวเตอร์อย่างถูกวิธี และด้วยคำสั่งใหม่เหล่านี้พวกเขาสามารถรับประกันได้ว่าโปรแกรมของพวกเขาจะทำงานจนจบลงอย่างเป็นปกติทุกครั้ง พวกเขาจะได้ไม่ต้องวิ่งกลับมาแก้ปัญหาเมื่อผู้ใช้ทำงานผิดพลาดเล็ก ๆ น้อย ๆ และจะได้มุ่งหน้าพัฒนาระบบงานอื่นต่อไป

ข้อเสียกลายเป็นข้อดี

ผู้ออกแบบภาษาเพื่อใช้เขียนโปรแกรมนั้นส่วนใหญ่เป็นนักวิชาการคอมพิวเตอร์ และเป็นนักวิชาการที่ไม่ค่อยมีประสบการณ์กับโลกธุรกิจ เขาออกแบบภาษาโปรแกรมขึ้นมาเพื่อให้ง่ายต่อการเข้าใจหลักการเบื้องต้น วัตถุประสงค์ก็เพื่อให้สามารถใช้สอนคนทั่วไปให้เขียนโปรแกรมได้รวดเร็ว ข้อเสียที่เห็นอยู่บ่อย ๆ คือใส่ข้อมูลเข้าไปผิดประเภท

แล้วมันจะหยุดกลางคัน แต่มันเหมาะกับการเขียนโปรแกรมโดยเจ้าของงานให้เจ้าตัวใช้งานเอง และถ้าจะให้ดีจริงก็จะต้องเป็นงานที่เข้าลักษณะเหมือนกับการใช้เครื่องคิดเลขและกระดาษทดเลข นั่นคือผู้เขียนโปรแกรมต้องมองให้เห็นว่าจานแม่เหล็กนั้นเหมือนกระดาษทดเลขหรือกระดาษทำการ ส่วนเครื่องคอมพิวเตอร์นั้นเป็นเครื่องคิดเลข แต่คำสั่งที่จะใช้กับคอมพิวเตอร์มันมีให้เลือกใช้ได้มากกว่าเครื่องคิดเลข มันจึงทำอะไรต่ออะไรได้มากกว่ากันเยอะเลย นอกจากนี้งานในคอมพิวเตอร์แบบนี้ต้องเป็นชนิดที่ทดลองแล้วทิ้งได้ คือถ้าเขียนโปรแกรมไม่ถูกหรือไม่ได้ผลตามต้องการก็ทำใหม่ได้ ถ้าทำได้ถูกต้องก็เอาไปใช้ได้ ใช้แล้วพร้อมที่จะทิ้งทั้งโปรแกรมและข้อมูล

ครับทั้งหมดนี้คือลักษณะการใช้คอมพิวเตอร์เมื่อ 40 ปีก่อนโน้น นักคณิตศาสตร์และนักวิทยาศาสตร์เป็นคนเขียนโปรแกรมเอง และเขาเป็นคนใช้งานเอง ถ้าโปรแกรมไม่ทำงานตามที่คิดไว้ก็ต้องแก้ปัญหา (debug) กันตรงหน้าเครื่องนั้นเลย เทคนิคนี้ต้องเลิกใช้ไปเมื่อเทคโนโลยีพัฒนาไปสู่เครื่องชนิดเมนเฟรม (Mainframe) ซึ่งผู้เขียนโปรแกรมต้องออกไปอยู่นอกห้องคอมพิวเตอร์ ส่วนผู้ใช้ระบบกลับเป็นคนในฝ่ายงานอื่น แต่ภาษาเพื่อเขียนโปรแกรมก็มิได้พัฒนาให้ทันกับเหตุการณ์เท่าใดนัก

อย่างไรก็ตาม โลกของเราได้

พัฒนาย้อนไปสู่ยุคต้น นั่นคือเราได้เครื่องคอมพิวเตอร์ขนาดเล็กและราคาถูกจนเราสามารถซื้อมาใช้เองบนโต๊ะทำงานได้ เมื่อเป็นเช่นนี้ความคิดที่จะใช้คอมพิวเตอร์ในแบบดั้งเดิม (หรือยุคบุกเบิก) ก็เป็นเรื่องที่เป็นไปได้ และน่าที่จะได้รับการส่งเสริมด้วย ทั้งนี้เพราะเรามีงานจำนวนมากมายที่เข้าลักษณะใช้ครั้งเดียวทิ้ง งานเหล่านี้ควรให้เจ้าของงานเขียนโปรแกรมเองและใช้เอง งานแบบนี้เจ้าของงานจะคิดทำอะไรก็ทำได้เลย จะเปลี่ยนแปลงแก้ไขเพิ่มเติมอะไรก็ได้ ไม่ต้องเสียเวลาถ่ายทอดความต้องการไปสู่โปรแกรมเมอร์ ข้อเสียของภาษาจึงกลายมาเป็นข้อดีอีกวาระหนึ่ง

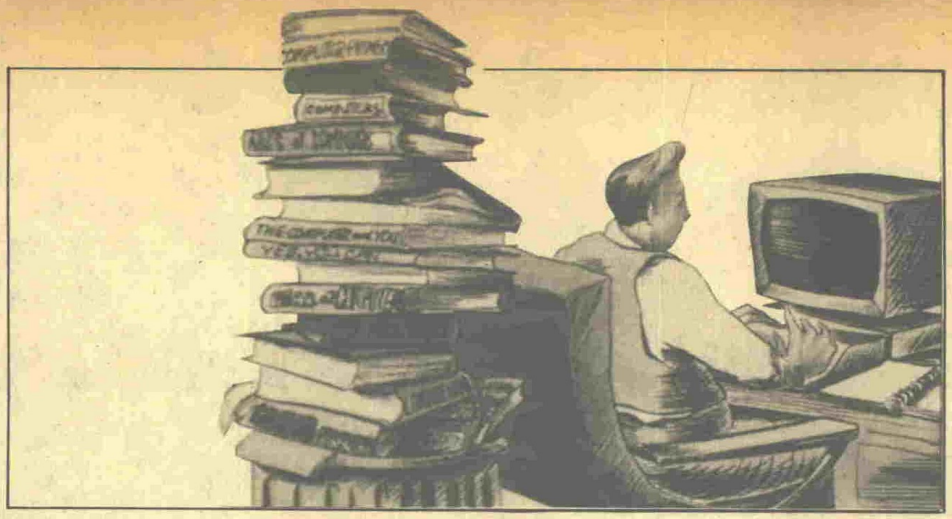
ตัวอย่างงานที่เหมาะสมเมื่อจะเขียนโปรแกรมเอง

เรื่องของการผ่อนส่งนั้นย่างกรายเข้ามาในชีวิตคนอย่างหลีกเลี่ยงไม่ได้ คำถามที่พบอยู่เสมอ ๆ คือ เวลาที่เราต้องการซื้อสินค้าชิ้นหนึ่ง แต่เรายังไม่อยากจ่ายครั้งเดียวหมด เราจะต้องผ่อนส่งเป็นงวด ๆ บางอย่างงวดละ 1 เดือน บางอย่างงวดละ 6 เดือน เรารู้อัตราดอกเบี้ย เรามีเงินก้อนเล็ก ๆ จ่ายก่อนแล้วก็ถามว่าที่เหลือจะต้องจ่ายงวดละเท่าใด ลูกค้าต้องการยืดหดจำนวนงวดดู ทั้งนี้เพื่อปรับให้เหมาะกับกำลังเงิน เรื่องนี้ถ้าจะต้องไปเปิดตารางอาจปวดเศียรเวียนเกล้า แถมยังคำนวณจำนวนเงินงวดสุดท้าย (ซึ่งมักจะไม่เท่ากับงวดอื่น ๆ) ได้ยากด้วย งานอย่างนี้เขียนโปรแกรมประเดี๋ยวเดียวก็เสร็จ และไม่ต้องไปหาซื้อโปรแกรมสำเร็จรูปมาใช้ให้สิ้นเปลืองและยุ่งยาก

งานคำนวณหาต้นทุนต่อหนึ่งหน่วยเป็นอีกงานหนึ่งที่ต้องใช้อยู่เสมอ ๆ ค่าใช้จ่ายนั้นมี 2 ประเภทใหญ่ ๆ คือ ประเภทคงที่กับประเภทแปรผันไปตามจำนวนการผลิต สูตรในการคำนวณนั้นง่ายมาก เขียนโปรแกรมยาวแค่สิบกว่าบรรทัดก็ใช้ได้แล้ว ถ้าจะให้ดียิ่งขึ้นเราอาจแยกประเภทของค่าใช้จ่ายให้ชัดเจนกว่าเดิมได้ ส่วนค่าใช้จ่ายประเภทแปรตามจำนวนการผลิตนั้นจะปรับให้เป็นเส้นโค้งเล็กน้อยก็ทำได้ หรืออาจจะคิดคำนวณถึงรายได้จากการเอาเศษวัสดุและของเสียหายไปขายอีกด้วย เรื่องเหล่านี้สามารถต่อเติมได้โดยง่าย และจะทำให้มีความละเอียดรอบคอบดีกว่าเก่ามาก

ต่อจากการคำนวณต้นทุนการผลิต เราอาจใช้คอมพิวเตอร์เพื่อจำลองแบบหรือจำลองเหตุการณ์ (Simulation) เรื่องนี้ไม่จำเป็นต้องไปเรียกใช้โปรแกรมสำเร็จรูปให้ยุ่งยาก เราใช้วิธีเปรียบเทียบระหว่างการเขียนโปรแกรมสั้น ๆ (200-500 บรรทัด) กับการศึกษาคำสั่งในโปรแกรมสำเร็จรูปแล้วเอามาทำเองอาจเสร็จสิ้นได้เร็วและตรงเป้ากว่าซะอีก

นอกจากงานคำนวณต่าง ๆ ที่กล่าวมาแล้ว การเก็บข้อมูลเป็นแฟ้ม



ขนาดเล็กแล้วเรียกมาใช้เป็นครั้งคราว ก็ทำโปรแกรมได้ไม่ยาก จะทำรายงานโดยจัดเรียงแบบไหน จะเพิ่มเติมข้อมูลอื่น ๆ ก็ทำเองหรือแก้ไขเองได้

ข้อควรระวัง

การเขียนโปรแกรมขึ้นมาใช้เองนี้มีความคล่องตัวดีมาก แต่ความคล่องตัวนี้ได้มาจากการ "ไม่ต้องทำเอกสาร (documentation)" และการ "แก้ไขได้ตามอำเภอใจ" ซึ่งทั้งสองสิ่งนี้จะไม่เหมาะกับงานที่เข้าลักษณะเป็นงานประจำ งานประจำนั้นหยุดไม่ได้ เมื่อผู้เขียนโปรแกรมโยกย้ายงานไปตำแหน่งอื่นก็ต้องส่งมอบให้คนอื่นดูแลต่อไป เมื่อเป็นเช่นนี้จึงเป็นเรื่องสำคัญอย่างยิ่งที่จะต้องระวังและหลีกเลี่ยงการใช้งานประเภทดังกล่าว หรือพูดง่าย ๆ คือถามตนเองอยู่เสมอ ๆ ว่า "งานนี้หยุดได้ไหม?"

อย่างไรก็ตาม เราไม่ควรจำกัดการใช้จนเกินไปนัก ควรยอมให้นำโปรแกรมไปใช้กับงานสำคัญ ๆ ได้ด้วย แต่เราต้องทำเอกสารเอาไว้บ้าง และพยายามลดการแก้ไขตามอำเภอใจ ในกรณีเช่นนี้ควรให้ฝ่ายตรวจสอบได้เข้ามาตรวจเป็นครั้งคราว

ข้อควรระวังอีกอย่างหนึ่งก็คือ ถึงแม้ว่าจะเป็นคนเขียนโปรแกรมเองและใช้เอง แต่อย่าปักใจเชื่อว่าความจำของเราจะดีเสมอไป ผมเป็นคนหนึ่งที่เขียนโปรแกรมให้ตนเองใช้เป็นจำนวนมาก และผมพบว่าโปรแกรมที่เราไม่ได้หยิบมาใช้งานเกิน 3 เดือน เราจะลืมเกือบหมด โดยเฉพาะอย่างยิ่งข้อมูลที่เก็บไว้ในแผ่นดิสเกตต์หลาย ๆ แผ่นนั้น เราจะสับสนได้ง่าย แฟ้มข้อมูลมักมีชื่อเดียวกัน แต่ตัวเนื้อข้อมูลคนละอย่าง เช่น เติมฟิลด์บางอย่างเข้าไปแล้ว แฟ้มใหม่จะใช้กับโปรแกรมเก่าไม่ได้ ชื่อของโปรแกรมอาจมีเก็บไว้ซ้ำ ๆ เป็นสิบ ๆ รุ่น แต่ละรุ่นมีการแก้ไขปรับปรุงแตกต่างกัน มันจึงทำงานไม่เหมือนกัน เรื่องนี้ต้องยอมเสียเวลาทบทวนสักหน่อย และถ้าจะให้ดีต้องทำบันทึกเอาไว้ ผู้อ่านลองนึกดูเอาเองก็ได้ว่า คนเขียนโปรแกรมมา 3-4 ปีจะมีดิสเกตต์อยู่ 5 ถึง 10 กล่อง แต่ละกล่องมี 10 แผ่น แต่ละแผ่นมีแฟ้มข้อมูลหรือโปรแกรม 20-30 ชื่อ ใครจะจำได้หมด มันจึงเป็นสิ่งจำเป็นอย่างยิ่งที่จะต้องบันทึกช่วยจำเอาไว้บ้าง

บทสรุป

งานเขียนโปรแกรมให้คนอื่นใช้งานนั้นยุ่งยากกว่าที่เราคิด ภาษาเพื่อใช้เขียนโปรแกรมนั้นออกแบบมาเพื่อให้คนเข้าใจง่ายและหัดเขียนโปรแกรมได้ใน 2-3 วัน แต่ต้องเป็นโปรแกรมที่เขียนให้ตนเองใช้เท่านั้น มันจึงเป็นปัญหากับนักเขียนโปรแกรมเป็นอย่างยิ่ง งานเขียนโปรแกรมได้กลายเป็นงานอาชีพเฉพาะ ผู้ใช้ต้องเรียนรู้ข้อบกพร่องของภาษา แล้วพยายามแสวงหาวิธีการที่จะทำให้โปรแกรมทำงานโดยไม่ผิดพลาด แต่เราก็แก้ปัญหาได้ไม่หมด เพราะเราไม่สามารถพัฒนาคนเขียนโปรแกรมให้ทันกับความต้องการของตลาด เราจำเป็นต้องหันกลับมาใช้แรงงานของฝ่ายเจ้าของงานเป็นผู้เขียนโปรแกรมช่วยบ้าง เราโชคดีมากที่มีคนพัฒนาเครื่องประเภทไมโครคอมพิวเตอร์ หรือคอมพิวเตอร์ส่วนบุคคลขึ้นมา เครื่องแบบนี้เหมาะสำหรับให้ยูสเซอร์เขียนโปรแกรมขึ้นมาใช้งานเอง และเขาสามารถใช้คำสั่งที่เป็นพื้นฐานได้ คิดอย่างไรก็เขียนไปตามนั้นไม่ต้องอ้อมค้อม โปรแกรมจะสั้นและทำงานได้อย่างรวดเร็ว มีข้อพึงระวังอยู่อย่างเดียวคือควรใช้กับงานชนิดที่จะเลิกใช้เมื่อใดก็ได้ หรือมองให้เห็นโดยแจ้งชัดว่าจานแม่เหล็กคือกระดาษทดเลขและซีพียูคือเครื่องคิดเลข ถ้าไม่มีเครื่องหรือคนเขียนโปรแกรมย้ายไปแล้วก็ให้ใช้แรงคนทำแทนได้

ผมเชื่อว่าหลังจากอ่านบทความนี้แล้วหลาย ๆ ท่านคงจะเข้าใจนักเขียนโปรแกรมได้ดีขึ้น งานของเขาซับซ้อนกว่าที่คนธรรมดาเข้าใจกัน แค่คำสั่ง INPUT A คำเดียวต้องเปลี่ยนเป็นโปรแกรมยาวเกือบ 2 หน้ากระดาษ แล้วมันยังมีเรื่องคล้าย ๆ กันนี้อีกนับร้อยนับพันเรื่อง กว่าจะเรียนรู้ได้หมดก็ต้องเหนื่อยอยู่หลายปี ผู้อ่านที่มีหัวทางลอจิกดี ๆ น่าจะหันมาช่วย ทั้งนี้โดยการหัดเขียนโปรแกรมให้ตนเองใช้ดูบ้างแล้วท่านจะสนุกกับงานยิ่งขึ้น